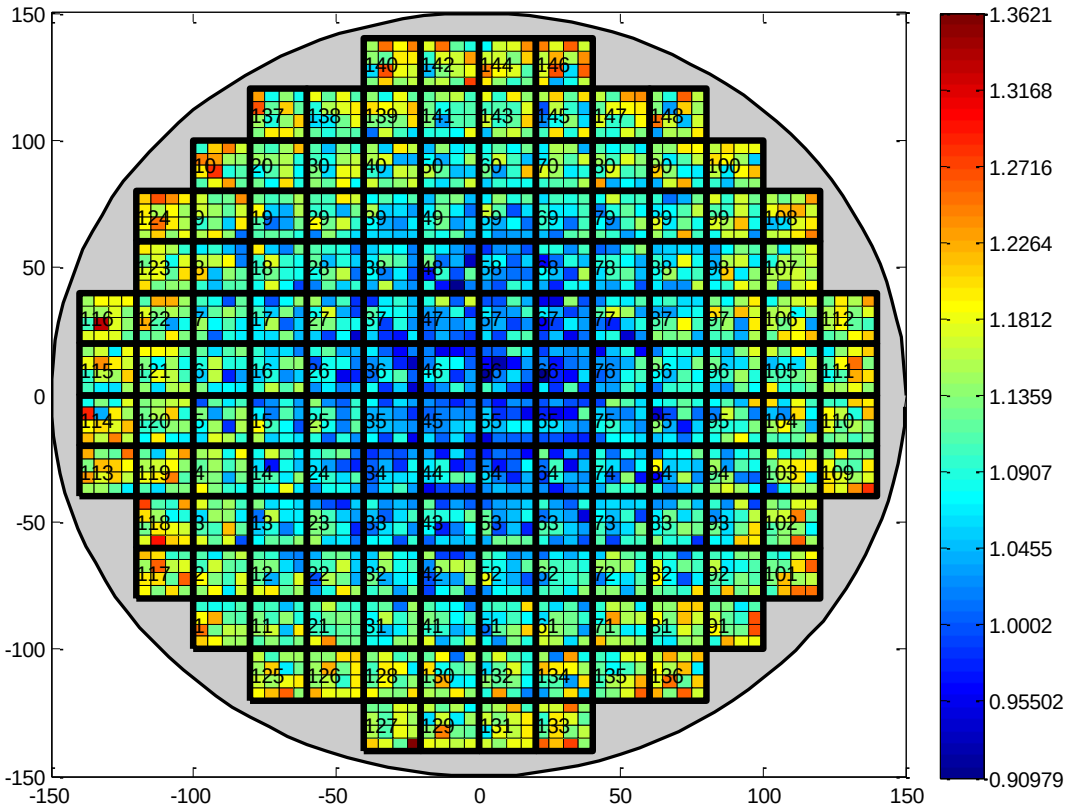
Report: Variance on a Wafer



Overview

Motivation:

Compare two Monte-Carlo Methods – the use of pseudorandom numbers compared to the used of quasi-Monte Carlo samples. These methods will be compared on a wafer-profit-model example, to see which method performs better, and to understand how to select the method for different applications.

Setup:

Using MATLAB, graphically model the placement of dies on a wafer.

Wafer diameter: 300

Die size: 1

Standard Deviation of Random Noise: 0.05

Grid Size on each die: 2x2=4

Systemic variation: (32/9000000) r2+ (1/1250) r + 1

Model the variance of the on each die of four values according to the sum of a parabolic systemic variation and randomly generated white noise. Then calculate the profit as a function of the lowest frequency values on each die and the resulting average profit. Examine the results of differently generated random noise.

Error:

We want to estimate our profit with the different variations. The profit function is:

$$13.53e^{2x}$$

where x is the minimum frequency on the die. The profit for a wafer with no variation is called the **nominal value**. This value is greater than the value with variations and a higher deviation results in lower profit values. The actual profit value is **estimated** using the Monte Carlo methods, and in these methods, an increasing number of samples improves the accuracy of the estimate. The difference between the estimate and the actual value is the **error** in the Monte Carlo estimation.

Monte Carlo:

This term refers to a group of computational algorithms that use pseudorandom sampling to model results. Pseudorandom sequences are generated using an algorithm to mimic the properties of true random numbers. For example, they exhibit the behavior of no patterns or regularities. In contrast with true random sequences however, pseudo random sequences are generated via a deterministic procedure, which utilize a "seed" to generate the sequence of numbers. Such a process is used because it is impossible to generate pure random numbers on a standard computer. A Monte Carlo algorithm generally follows these steps:

1. Define a domain of possible inputs.
2. Randomly generate input values from the domain with a specified probability distribution.
3. Perform a deterministic computation using the inputs.
4. Combine the results of the computations as a final result, which is an approximation.

Example: The Monte Carlo method can be used to approximate the value of pi.

The domain is the two dimensional points bound by the lines

X=0, y=0, x=1, y=1

Then we generate a pseudorandom stream of points with a uniform distribution over the domain. We want to find the number of points that lie in a quarter circle with the center at the origin, so we test each point (x ,y) to see if $x^2+y^2<1$. The ratio of the number of points inside the circle to the total number of points will approach the ratio of the area of the circle to the area of the square. Thus, the value of pi would be simply approximated as

4(Number of Points Inside the Circle)/(Total Number of Points Generated)

In the case of the wafer, what we attempt to approximate is the average profit with a set variance as the sum of a random sequence and a systemic sequence. We observe the change in error as more inputs are used.

Quasi Monte Carlo:

The difference of QMC from MC is that QMC relies on low discrepancy sequences to generate its inputs while MC relies on pseudorandom sequences. This means that overall QMC can be expected to yield a better accuracy than MC for the same number of inputs. However, because QMC relies on the concept of low discrepancy, the number of samples needs to be greater than the dimension to be accurate. In high dimensions, many points are required to give the QMC and advantage over the MC. Unlike MC however, the samples are not randomly dispersed but are related in a way to create low discrepancy (uniform distribution) among the sequences. This means that enough points must be generated to create low discrepancy, and the number of points is proportional to the exponential of the dimension.

Data

Calculated Average Profit (tens of thousands of dollars) Dimensions: 2368

# of Trials Type	5	10	15	20	25	30	35	40	45	50	10000
MC	1.5430	1.5385	1.5375	1.5420	1.5419	1.5400	1.5411	1.5402	1.5403	1.5405	1.5405
QMC	1.5360	1.5361	1.5368	1.5373	1.5376	1.5381	1.5387	1.5391	1.5396	1.5401	1.7170

Percentage Error (%) Dimensions: 2368

Type\# of Trials	5	10	15	20	25	30	35	40	45	50
MC	0.1623	0.1298	0.1947	0.0974	0.0909	0.0325	0.0389	0.0195	0.0130	0
QMC	13.8445	9.9555	8.0301	6.0237	5.2881	4.0893	3.1108	2.4045	2.1099	1.5655

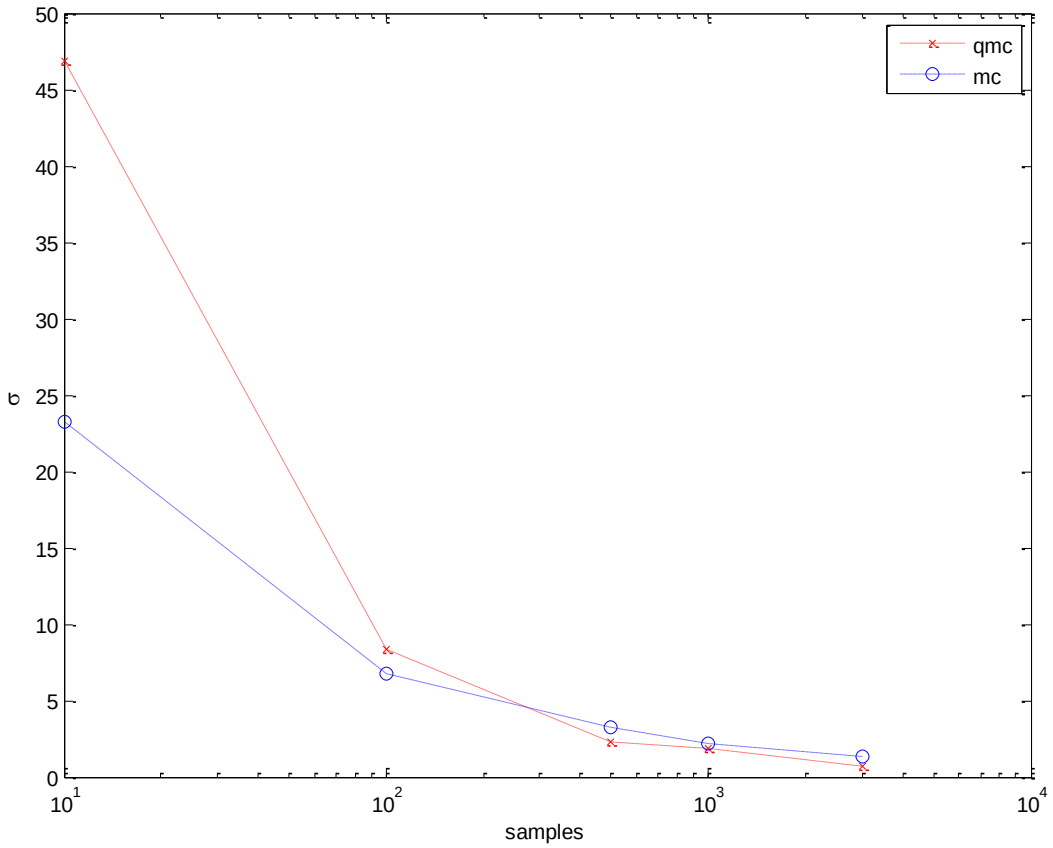
Dimensions: 2368

	Sigma 10	Sigma 100	Sigma 1000
MC	7.1353	6.7691	2.3409

Quasi Monte Carlo does not converge quickly enough for 2368 dimensions. In this case, it is preferable to run Monte Carlo estimation over Quasi-Monte Carlo. For a reduced dimension of 35 (Changed the die size: 4, and Grid Size: 1x1=1) the analysis is:

Dimensions: 35

	Sigma 10	Sigma 100	Sigma 500	Sigma 1000	Sigma 3000
MC	23.2779	6.7813	3.2406	2.2104	1.3634
QMC	46.8910	8.2996	2.3315	1.9111	0.7155



Research

How are the Halton sequences of quasi random numbers generated?

Halton sequences all share the characteristic of low discrepancy and are deterministic, yet appear to be random for computation algorithms. They are a subset of quasi-random numbers. Every Halton sequence, regardless of dimension, use prime number(s) for its base(s). In J. H. Halton's paper on Halton sequences (1958?), he proposes the following:

Express any integer n as the sum of the successive powers of radix R. That is,

$$n = n_M n_{M-1} \ldots n_2 n_1 n_0 = n_0 + n_1 R + n_2 R^2 + \ldots + n_M R^M, \text{ where } M = [\log_R n]$$

Then, a new fraction, f, between 0 and 1, is formed when all the powers of the radix are changed with their respective inverses. That is,

$$f = f_R(n) = 0.n_0\, n_1\, n_2 \ldots n_M\, n_{M-1} = n_0 R^{-1} + n_1 R^{-2} + n_2 R^{-3} + \ldots + n_M R^{-M-1}$$

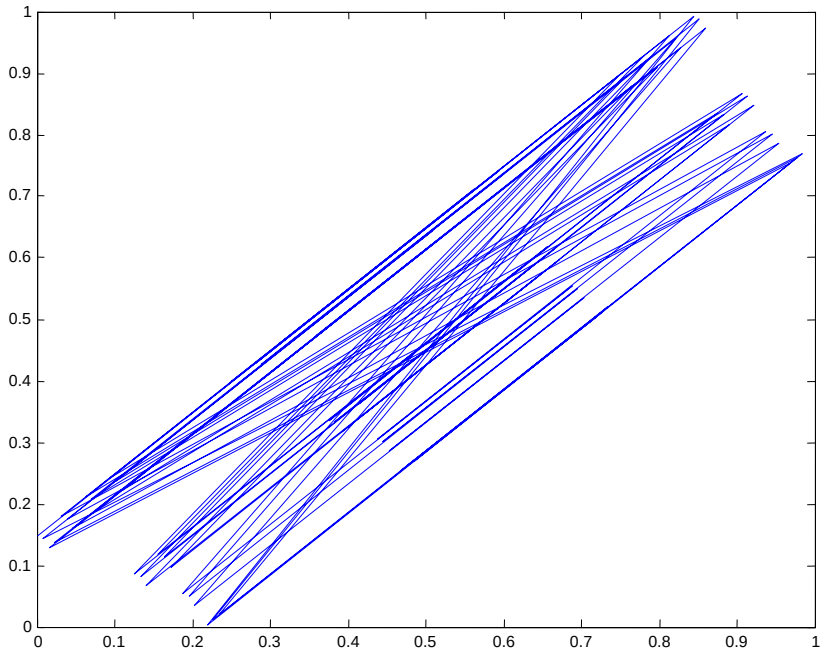
To generate N numbers in K dimensions, use the k-dimensional space

$$(n/N,\ f_{R1}(n),\ f_{R2}(n),\ \ldots,\ f_{RK-1}(n)),$$

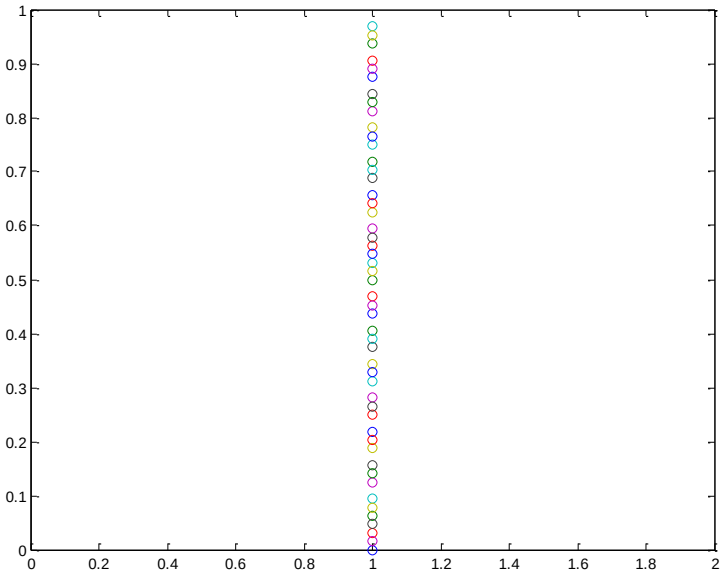
where n = 1, 2, …, N and $R_1, R_2, \ldots, R_{k-1}$ are the first k-1 primes.

Why is it crucial to generate correct dimensions of Halton sequences?

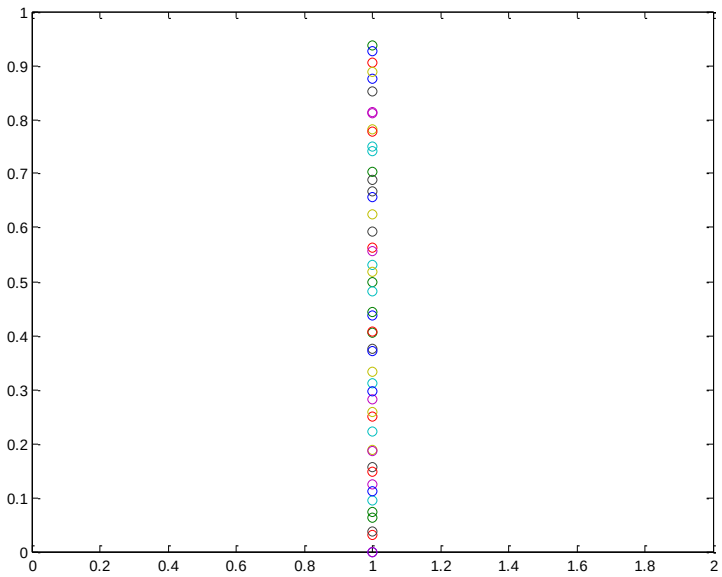
If lower dimension Halton sequences are spliced and paired up to form higher dimension sequences, the linear correlation of the identical radices will form a patterned plot of points, eliminating the desirable low discrepancy characteristic.



If higher dimension Halton sequences are spliced and lined up to form lower dimension sequences, the change of radices partway results in more than one low discrepancy sequence. Thus, the combined sequence is not as low discrepancy because the positions of the values of one radix does not “know” where the positions of the values of another radix is, as in the case where the Halton sequence is generated with the correct dimension(s).



1 dimension sequence in 1 dimension



2 dimension sequence in 1 dimension

Why is it not viable to use a spaced grid of numbers as inputs?

The amount of work to obtain the same amount of precision is independent of the dimension *d* of the underlying random variables.

Thus Monte Carlo integration is practically the only method to numerically compute high-dimensional integrals. Traditional quadrature techniques generally require an amount of work exponential in the number of dimensions *d* , since they require sampling a grid in *d* -dimensional space.

On the other hand, Monte Carlo integration is generally not competitive with quadrature for low-dimensional integration (e.g. *d*=1 or *d*=2).